GESTIÓN DE USUARIOS EN MYSQL

ÍNDICE DE CONTENIDO

1. GESTIÓN DE USUARIOS

MySQL presenta un esquema de seguridad refinado, flexible y basado en la lista de control de accesos.
Un usuario en MySQL se identifica por el login e identificador del servidor cliente (IP, nombre), esto se basa en el principio de que el usuario que se conecta desde la oficina no tiene porqué ser el mismo que se conecta desde la casa.

En gestores como Oracle, la cosa se complica algo más. Si quieres saber cómo se realiza en Oracle puedes consultar este manual:

https://orasite.com/tutoriales/administracion/administracion-de-usuarios-en-oracle

Veremos en esta unidad cómo crear y manipular usuarios y permisos.

1.1 CREAR USUARIOS

Para agregar un nuevo usuario es tan simple como ejecutar esta sentencia:

```
CREATE USER 'nombre_de_usuario'@'host' IDENTIFIED BY 'clave en texto plano';
```

El “nombre de usuario” es el identificador del perfil o de la persona que quiere conectarse a la BD y el “host” es la máquina/equipo/servidor (dirección IP) desde la que se quiere conectar por lo que, realmente, no estamos creando un usuario sino un perfil de conexión de un usuario (usuario+host).

Generalmente, al crear usuarios en MySQL especificaremos el host del usuario como localhost para prevenir conexiones desde hosts no deseados. Sólo en casos muy específicos cambiaremos ese “localhost” por una IP de una máquina concreta o el comodín “%” que veremos más adelante.

Es importante que entiendas que el host se refiere a la máquina desde donde se conecta el usuario a la base de datos. Por ejemplo, si el usuario se sentará físicamente en el mismo portátil o PC donde está la base de datos instalada, deberíamos crear un perfil de conexión de ese usuario para localhost.

Por ejemplo, para agregar al usuario andergarcia con la clave 123456 conectado desde la misma máquina donde está alojada la base de datos (localhost), se utilizaría este comando:

```
CREATE USER 'andergarcia'@'localhost' IDENTIFIED BY '123456';
```

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se conecta desde la misma máquina de la base de datos (localhost).

Sin embargo, si queremos especificar permisos para cuando ese mismo usuario se conecta desde una IP determinada, por ejemplo desde su casa, podemos especificarlo de la siguiente forma:

```
CREATE USER 'andergarcia'@'252.21.12.2' IDENTIFIED BY '123456';
```

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se conecta desde su casa (252.21.12.2).

También podemos asignar permisos a un usuario concreto cuando se conecte desde cualquier parte, desde de cualquier máquina, usando el comodín %:

```
CREATE USER 'andergarcia'@'%' IDENTIFIED BY '123456';
```

En el caso anterior, estaríamos creando un perfil de conexión del usuario andergarcia para cuando se

conecta desde cualquier máquina (%).

Piensa que cada nombre de usuario se almacena junto a su host en una tabla llamada user de una de las bases de datos del sistema llamada mysql, y ambos campos son clave primaria.
Con esto, cuando creamos un nuevo usuario estamos creando realmente un usuario y un host o, como lo hemos nombrado al principio, un perfil de conexión de un usuario (usuario+host).

Quizás lo entiendas mejor si accedes a la BD de tu equipo y muestras la estructura de la mesa que te comentamos. Recuerda: mesa user de la BD interna mysql. Sería algo así:

Como ves, la clave primaria compuesta nos permite repetir cuántas veces queramos el nombre de usuario y el host pero nunca repetir la misma combinación de ambos. Si esto no te queda claro, revisa los temas anteriores antes de seguir.

> IMPORTANTE
>
> Todos los usuarios (perfiles de conexión de usuarios) y los permisos, que veremos a continuación, se almacenan en una misma tabla llamada user de la base de datos interna mysql, en el servidor MySQL.

Porque lo veas más claro. Piensa que en estas tres sentencias estamos creando realmente TRES USUARIOS (usuario+host) diferentes a los que asignaremos permisos diferentes:

– Ander desde su casa:

```
CREATE USER 'andergarcia'@'254.68.2.15' IDENTIFIED BY '123456';
```

– Ander desde la oficina

```
CREATE USER 'andergarcia'@'214.168.42.60' IDENTIFIED BY '123456';
```

– Ander desde la misma máquina donde está instalada la BD

```
CREATE USER 'andergarcia'@'localhost' IDENTIFIED BY '123456';
```

1.2 CAMBIAR PASSWORD

Si más adelante quisiste cambiar (o asignar en caso de inexistencia) una contraseña a un determinado usuario+host, se utilizará ALTER USER como se muestra a continuación:

```
ALTER USER 'nombre_de_usuario'@'host' IDENTIFIED BY 'nueva clave en texto plano';
```

Por ejemplo, para cambiar la clave de acceso cuando Ander se conecta desde la oficina:

```
ALTER USER 'andergarcia'@'214.168.42.60' IDENTIFIED BY '987654';
```

En el caso anterior, cambiaríamos ÚNICAMENTE la clave del usuario andergarcia, pero sólo cuando se conecta desde esa dirección IP, dejando el resto de claves (desde otros host) intactas.

1.3 ELIMINAR USUARIOS

Por otra parte, al igual que puedes eliminar bases de datos y tablas con DROP, también puedes usar DROP USER para eliminar un usuario+host por completo:

```
DROP USER 'nombre de usuario'@'host';
```

Por ejemplo, para eliminar el usuario+host de Ander conectado desde la oficina haremos:

```
DROP 'andergarcia'@'214.168.42.60';
```

En el caso anterior, estaríamos eliminando ÚNICAMENTE el perfil de conexión del usuario andergarcia, pero sólo cuando se conecta desde esa dirección IP, dejando el resto de perfiles de conexión (desde otros host) intactos.

1.4 INICIAR SESIÓN CON UN NUEVO USUARIO

Para iniciar sesión con un usuario concreto usa este comando desde WorkBench:

Nos pediría la clave:

No te asustes si ves la palabra ORACLE en pantalla. La empresa MySQL AB fue adquirida por la empresa Sun Microsystems en 2008 y ésta fue comprada por la empresa Oracle Corporation en 2010. Sin embargo, los productos SGBD MySQL y SGBD Oracle son diferentes y 100% independientes, ambos de tipo relacional.

Recuerda pues que Oracle es una empresa y el SGBD MySQL y el SGBD Oracle son productos de esa misma empresa.

Existe una versión “libre” del SGBD MySQL llamada MariaDB, también relacional y muy similar al SGBD MySQL. MariaDB es un SGBD relacional muy similar a MySQL, puesto que fue creado por uno de los desarrolladores de MySQL. El objetivo de su desarrollo fue mantener el software de gestión de base de datos en un modelo de software libre.

Más info:

- https://www.hostingplus.com.es/blog/que-es-mariadb-y-cuales-son-sus-caracteristicas/

2. PERMISOS DE USUARIOS SOBRE RECURSOS

2.1 OTORGAR PERMISOS

Volviendo al punto 1, cuando creamos un usuario+host con la orden CREATE USER, ese usuario no tiene

acceso a nada. De hecho, incluso si ese usuario intenta iniciar sesión (con la contraseña indicada), ¡sólo podrá conectar en MySQL!

Por tanto, lo siguiente que haremos después del CREATE USER es dar a ese usuario+host acceso a la información que necesitará, con esta sintaxis:

```
GRANT ALL PRIVILEGES DONDE 'basededatos'.'tabla' TO 'nombre_de_usuario'@'host';
```

Por ejemplo, para agregar permisos al usuario +host andergarcia sobre todas las bases de datos y todas sus tablas se utilizaría este comando:

```
GRANT AJO PRIVILEGAS DONDE *.* TO 'andergarcia'@'localhost';
```

Los asteriscos, en este comando, se refieren a la base de datos y las tablas (respectivamente) a los que pueden acceder. Este comando específico permite al usuario leer, editar, ejecutar y realizar todas las tareas en todas las bases de datos y tablas.

En el caso anterior, estaríamos dando TODOS LOS PRIVILEGIOS al usuario andergarcia sobre TODAS LAS BASES DE DATOS y TODAS SUS TABLAS cuando accede desde localhost.

> IMPORTANTE
>
> Ten en cuenta que en este ejemplo estamos otorgando a andergarcia acceso total a toda nuestra base de datos cuando se conecta desde la misma máquina en la que está instalada la base de datos. Si bien esto es útil para explicar algunos conceptos de MySQL, puede ser poco práctico para la mayoría de casos de uso y podría poner la seguridad de tu base de datos en alto riesgo.

Una vez finalizados los permisos que quieres configurar para tus nuevos usuarios, asegúrate siempre de “refrescar” todos los privilegios mediante este comando:

```
FLUSH PRIVILEGAS;
```

En algunos escenarios no es necesaria el orden anterior, pero si lo ejecutas siempre te aseguras que los cambios se aplican correctamente (buena praxis).

Tras aplicar esa orden, los cambios ahora estarán vigentes.

Como dijimos, usar ALL PRIVILEGES puede ser demasiado amplio y puede poner en riesgo la seguridad de tu servidor de base de datos por lo que debes evitar conceder ese permiso a ningún usuario+host sin haberlo meditado y justificado antes. Lo mismo ocurre con @'%'.

Podríamos hacer una analogía con el uso del SELECT * FROM:

¿Podemos usarlo siempre? SI. ¿Es siempre la mejor opción? NO.

Éstos son los posibles permisos comunes que puedes conceder a un usuario+host.

- AJO PRIVILEGAS: Como vimos antes, esto le otorgaría a un usuario+host acceso completo a una o varias bases de datos.
- CREATE: Permite crear nuevas tablas o bases de datos.
- DROP: Permite eliminar tablas o bases de datos.

- DELETE: Permite eliminar filas de las tablas.
- INSERT: Permite insertar filas en las tablas.
- SELECT: Les permite usar el comando SELECT para leer las bases de datos.
- UPDATE: Permite actualizar las filas de las tablas.
- WITH GRANT OPTION: Permite otorgar o eliminar privilegios de otros usuario+host. Cuando otorgamos permisos a un determinado usuario+host, es posible además conceder un permiso extra que le permita asignar sus mismos privilegios a otros usuario+host. El perfil ALL PRIVILEGES no incluye GRANT OPTION, por lo que habría que dárselo posteriormente.

Para proporcionar un permiso específico, puede usarse este en sintaxis:

```
GRANT permisos ON basededatos.tabla TO 'nombre_de_usuario'@'host' [WITH GRANT OPTION];
```

Si quieres dar a un usuario+host acceso a cualquier base de datos oa cualquier tabla, asegúrate de poner un asterisco (*) en el lugar del nombre de la base de datos o de la tabla.

¡Ah! y cada vez que actualices o cambias un permiso, asegúrate de usar FLUSH PRIVILEGES.

Por ejemplo, si queremos dar los permisos de selección y actualización sobre una determinada tabla a un determinado usuario cuando se conecta desde la propia máquina de la base de datos, y otros diferentes si se conecta desde fuera haríamos algo como:

```
GRANT SELECT, INSERT, UPDATE, DELETE DONDE ciclismo.ciclista TO 'andergarcia'@'localhost';
GRANT SELECT, INSERT, UPDATE DONDE ciclismo.ciclista TO 'andergarcia'@'192.168.2.6';
GRANT SELECT DONDE ciclismo.* TO 'andergarcia'@'%';
```

Aunque no es tan común, y quizás no tan legible, también podemos aplicar varios permisos, a varios perfiles… todo en una misma orden. Lo que no puedes hacer es asignar esos permisos a varios recursos a la vez. De esta forma:

```
GRANT permiso, permiso, permiso...
 DONDE basededatos.mesa, basededatos.mesa, basededatos.mesa
 TO 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host' , 'nombre_de_usuario'@'host';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
DONDE ciclismo.*
TO 'andergarcia'@'localhost', 'andergarcia'@'214.168.42.60', 'andergarcia'@'254.68.2.15';
```

2.2 REVOCAR PERMISOS

Por otra parte, si necesitas revocar un permiso, ésta es la estructura:

```
REVOKE permisos ON basededatos.tabla FROM 'nombre_de_usuario'@'host';
```

> IMPORTANTE
>
> Ten en cuenta que, cuando revocas permisos, la sintaxis requiere que utilices FROM en lugar de TO, como se utiliza para otorgar permisos.

Por ejemplo, para revokar los permisos concedidos al usuario andergarcia, se utilizaría:

```
REVOKE ALL PRIVILEGES DONDE *.* FROM 'andergarcia'@'localhost';
```

Cuando otorgas WITH GRANT OPTION a un usuario+host, le dejas dar (y después revocar) los permisos que le das (todos o parte de ellos) a CUALQUIER USUARIO.

Por ejemplo, para estos usuarios

- [root] CREATE USER piedra; -- (se crea pedro@%' sin clave)
- [root] CREATE USER luís; -- (se crea luís@'%' sin clave)
- [root] CREATE USER laura; -- (se crea laura@'%' sin clave)
- [root] CREATE USER sara; -- (se crea sara@'%' sin clave)

Debes tener en cuenta que, cuando no se especifica el host, éste se asume por defecto como @'%', y se refiere a la conexión desde cualquier PC del mundo.

Ahora, con estos usuarios creados:

Si otorgas 3 permisos sobre 1 mesa concreta a un usuario (ej, musica.club), este usuario podrá dar (y después revocar) cada uno de esos 3 permisos sobre esa mesa.

- [root] GRANT SELECT, INSERT, UPDATE DONDE musica.club TO piedra WITH GRANT OPTION;
- [root] QUIT
- [pedro] GRANT SELECT ON musica.club TO laura; -- (se asigna a laura@'%')
- [pedro] GRANT INSERT ON musica.club TO sara; -- (se asigna a sara@'%')
- [pedro] GRANT INSERT ON musica.club TO luís; -- (se asigna a lluís@'%')
- [pedro] REVOKE INSERT ON musica.club FROM lluís; -- (se revoca a lluís@'%')

Si otorgas 3 permisos sobre 1 base de datos entera a un usuario (ej, musica.*), este usuario podrá dar (y después revocar) cada uno de esos 3 permisos sobre todas las tablas de esa base de datos o, de forma detallada, sobre cada una de las tablas de música.

- [root] GRANT SELECT, INSERT, UPDATE ON musica.* TO piedra WITH GRANT OPTION;
- [root] QUIT
- [pedro] GRANT SELECT ON musica.club TO laura;
- [pedro] GRANT INSERT DONDE musica.cancion TO sara;
- [pedro] GRANT INSERT ON musica.* TO luís;

Por último, si otorgas TODOS LOS PERMISOS sobre todas las bases de datos a un usuario (ej, *.*), este usuario podrá dar (y después revocar) TODOS LOS PERMISOS sobre todas las bases de datos o, de forma detallada, sobre cada una de las bases de datos y sus tablas.

- [root] GRANT ALL PRIVILEGES DONDE *.* TO piedra WITH GRANT OPTION;
- [root] QUIT
- [pedro] GRANT SELECT ON musica.club TO laura;
- [pedro] GRANT INSERT ON musica.* TO sara;
- [pedro] GRANT INSERT ON ciclismo.* TO luís;

2.3 MOSTRAR PERMISOS

Por último, puedes revisar los permisos actuales de un usuario ejecutando lo siguiente:

```
SHOW GRANTES FOR 'nombre_de_usuario'@'host';
```

Por ejemplo, si queremos ver todos los permisos del usuario andergarcia deberemos pedir los permisos de cada una de las combinaciones usuario+host:

```
SHOW GRANTES FOR 'andergarcia'@'localhost';
```

2.4 ROLES DE USUARIO

MySQL 8.0 (19 April 2018) agrega administración de roles en la administración de usuarios, y también se ha ajustado el método de cifrado de contraseña predeterminado.SHA1 Cambiado aSHA2. Al mismo tiempo, con la función de MySQL 5.7 de deshabilitar a los usuarios y la caducidad de los usuarios, las funciones de administración de usuarios y la seguridad de MySQL se mejoran enormemente en comparación con la versión anterior.

No veremos gestión de roles este curso, al ser algo más propio de Oracle.

Si quieres más información en este enlace:

https://wiki.cifprodolfoucha.es/index.php?title=mysql_gesti%C3%B3n_de_permisos#ROLS

3. EJEMPLOS

3.1 EJEMPLO 1

Como ejemplo práctico, supongamos que tenemos dos tablas en la BD weblibros y esta consulta realizada por el usuario root (superadministador):

Después, ejecutamos esta consulta:

GRANT SELECT DÓNDE weblibros.categoría TO juanperez@localhost;

Si en vez del usuario root (superaministrador) la consulta “show tables” fuera realizada por el usuario juanperez, obtendríamos lo siguiente al carecer de permiso SELECT para ver todas las tablas:

Si el usuario juanperez intentara acceder a una mesa no permitida, el acceso le sería negado:

3.2 EJEMPLO 2

Suponiendo que en sistema existen las siguientes bases de datos y tablas:

```
CREATE DATABASE webcursos;

USE webcursos;

CREATE TABLE categoría(cat INT PRIMARY KEY);

CREATE TABLE libro(lib INT PRIMARY KEY);
```

Crearemos primero 3 usuarios, a los que sólo se les permita conectarse de forma local:

```
CREATE USER 'vanina'@'localhost' IDENTIFIED BY 'v4n1n41974';

CREATE USER 'roxana'@'localhost' IDENTIFIED BY 'r0x4n4-1275';

CREATE USER 'silvana'@'localhost' IDENTIFIED BY '33885409sil';
```

Y ahora, otorgaremos permisos de selección y actualización, a todos los usuarios recién creados, para todas las tablas de la base de datos weblibros:

```
GRANT SELECT, UPDATE ON weblibros.* TO 'vanina'@'localhost', 'roxana'@'localhost',
'silvana'@'localhost';
```

Ahora, al usuario vanina (desde localhost), le otorgaremos todos los permisos para la mesa categoría de la base de datos weblibro, permitiéndole conceder estos permisos a cualquier otro usuario existente:

```
GRANT ALL PRIVILEGES DONDE weblibros.* TO 'vanina'@'localhost' WITH GRANT OPTION;
```